

Sir:

00/22/90
JC490 U.S. PTO

Transmitted herewith for filing is the Patent Application of:

Inventor(s): Charles R. Moore

For: PROCESSOR AND METHOD OF EXECUTING A LOAD INSTRUCTION THAT BIFURCATE LOAD EXECUTION INTO
TWO OPERATIONS

Enclosed are:

Patent Specification and Declaration
 2 sheets of drawing(s).
 An assignment of the invention to International Business Machines Corporation (includes Recordation Form Cover Sheet).
 A certified copy of a application.
 Information Disclosure Statement, PTO 1449 and copies of references.



The filing fee has been calculated as shown below:

For	Number Filed	Number Extra	Rate	Fee
<u>Basic Fee</u>				<u>\$690.00</u>
Total Claims	16	- 20	x 18 =	\$
Indep. Claims	4	- 3	1	x 78 = \$ 78.00
<u>MULTIPLE DEPENDENT CLAIM PRESENTED</u>				x 260 = \$
				<u>TOTAL \$768.00</u>

Please charge IBM Corporation Deposit Account No. 09-0447 in the amount of \$768.00. A duplicate copy of this sheet is enclosed.
 The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to IBM Corporation Deposit Account 09-0447.
 Any additional filing fees required under 37 CFR §1.16.
 Any patent application processing fees under 37 CFR §1.17.

Respectfully submitted,

By Brian F. Russell
Brian F. Russell
Registration No. 40,796
FELSMAN, BRADLEY, VADEN, GUNTER & DILLON, LLP
Suite 350 Lakewood on the Park
7600B North Capital of Texas Highway
Austin, Texas 78731
Telephone (512) 343-6116

PROCESSOR AND METHOD OF EXECUTING A LOAD INSTRUCTION THAT
BIFURCATE LOAD EXECUTION INTO TWO OPERATIONS

5

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates in general to data processing and, in particular, to a processor and method of performing load operations in a processor. Still more particularly, the present invention relates to a processor and method of processing a load instruction that bifurcate load execution into two separate operations.

2. Description of the Related Art:

Most processors' instruction set architectures (ISAs) include a load or similar type of instruction that, when executed, causes the processor to load specified data from memory (e.g., cache memory or system memory) into the processor's internal registers. Conventional processors handle the execution of load instructions in one of two ways. First, a processor may execute load instructions strictly in program order. In general, the execution of load instructions with strict adherence to program order is viewed as disadvantageous given the fact that at least some percentage of data specified by load instructions will not be present in the processor's cache. In such cases, the processor must stall the execution of the instructions following the load until the data specified by the load is retrieved from memory.

Alternatively, a processor may permit load instructions to execute out-of-order with respect to the

programmed sequence of instructions. In general, out-of-order execution of load instructions is viewed as advantageous since operands required for execution are obtained from memory as soon as possible, thereby improving overall processor throughput. However, supporting out-of-order execution of load instructions entails additional complexity in the processor's architecture since, to guarantee correctness, the processor must be able to detect and cancel an out-of-order load instruction that loads data from a memory location targeted by a later-executed store instruction (executed in the same or a remote processor) preceding the load instruction in program order.

5

10

ALL INFORMATION CONTAINED
HEREIN IS UNCLASSIFIED
DATE 10-12-2015 BY SP2000

SUMMARY OF THE INVENTION

5 The present invention addresses the poor performance associated with in-order processors and eliminates much of the complexity associated with out-of-order machines by providing an improved processor and method of executing load instructions.

10 In accordance with the present invention, a processor implementing an improved method for executing load instructions includes execution circuitry, a plurality of registers, and instruction processing circuitry. The instruction processing circuitry fetches a load instruction and a preceding instruction that precedes the load instruction in program order, and in response to detecting the load instruction, translates the load instruction into separately executable prefetch and register operations. The execution circuitry performs at least the prefetch operation out-of-order with respect to the preceding instruction to prefetch data into the processor and subsequently separately executes the register operation to place the data into a register specified by the load instruction. In an embodiment in which the processor is an in-order machine, the register operation is performed in-order with respect to the preceding instruction.

15 20 25

30 All objects, features, and advantages of the present invention will become apparent in the following detailed written description.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

10

Figure 1 depicts an illustrative embodiment of a data processing system with which the method and system of the present invention may advantageously be utilized;

15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95

Figure 2A and **2B** illustrate two alternative embodiments of the translation of UIISA load instructions into separately executable PREFETCH and REGISTER operations in accordance with the present invention; and

Figure 3 is an exemplary load data queue that may be utilized to temporarily buffer load data in accordance with the present invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENT

With reference now to the figures and in particular with reference to **Figure 1**, there is illustrated a block diagram of an exemplary embodiment of a data processing system with which the present invention may advantageously be utilized. As shown, the data processing system includes at least one processor, indicated generally at **10**, which, as discussed further below, includes various execution units, registers, buffers, memories, and other functional units that are all formed within a single integrated circuit. Processor **10** is coupled by a bus interface unit (BIU) **14** to a bus **12** and other components of the data processing system, such as system memory **8** or a second processor **10** (not illustrated).

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

25

30

Processor **10** includes an on-chip multi-level cache hierarchy **16** that provides low latency access to cache lines of instructions and data that correspond to memory locations in system memory **8**. In the depicted embodiment, cache hierarchy **16** includes separate level one (L1) instruction and data caches **13** and **15** and a unified level two (L2) cache **17**. An instruction sequencing unit (ISU) **20** requests instructions from cache hierarchy **16** by supplying effective addresses (EAs) of cache lines of instructions. In response to receipt of an instruction request, cache hierarchy **16** translates the provided EA into a real address and outputs the specified cache line of instructions to instruction translation unit **18**. Instruction translation unit **18** then translates each cache line of instructions from a user instruction

5

10

set architecture (UISA) into a possibly different number of internal ISA (IISA) instructions that are directly executable by the execution units of processor 10. The instruction translation may be performed, for example, by reference to microcode stored in a read-only memory (ROM) template. In at least some embodiments, the UISA-to-IISA translation results in a different number of IISA instructions than UISA instructions and/or IISA instructions of different lengths than corresponding UISA instructions.

15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95

Following instruction translation by ITU 18, ISU 20 temporarily buffers the IISA instructions until the instructions can be dispatched to one of the execution units of processor 10 for execution. In the illustrated embodiment, the execution units of processor 10 include integer units (IUs) 24 for executing integer instructions, a load-store unit (LSU) 26 for executing load and store instructions, and a floating-point unit (FPU) 28 for executing floating-point instructions. Each of execution units 24-28 is preferably implemented as an execution pipeline having a number of pipeline stages.

25
30

During execution within one of execution units 24-28, an instruction receives operands (if any) from, and stores data results (if any) to one or more registers within a register file coupled to the execution unit. For example, IUs 24 execute integer arithmetic and logic instructions by reference to general-purpose register (GPR) file 32, and FPU 28 executes floating-point arithmetic and logic instructions by reference to floating-point register (FPR) file 34. LSU 26 executes load and store instructions to transfer data between

5

10

memory (e.g., cache hierarchy 16) and either of GPR file 32 and FPR file 34. After an execution unit finishes execution of an instruction, the execution unit notifies instruction sequencing unit 20, which schedules completion of the instruction in program order. Upon completion of an instruction, the data results, if any, of the instruction form a portion of the architected state of processor 10, and execution resources allocated to the instruction are made available for use in the execution of a subsequent instruction.

15

20
25

As noted above, much of the hardware and data flow complexity involved in processing load instructions in conventional processors is attributable to the execution of load and other instructions out-of-program order. In particular, the design philosophy of many conventional processors that permit out-of-order execution of instructions is to execute load instructions as early as possible to place specified data into a register file so that subsequent instructions having a dependency upon the load data are less likely to stall due to memory access latency. The processor must then detect data hazards (e.g., store instructions targeting the same address that are earlier in program order, but later in execution order) with respect to the data and discard the load data from the register file (and instructions executed utilizing that load data) in the event that the load data is found to be stale.

30

In accordance with the present invention, processor 10 simplifies the processing of UIISA load instructions by translating at least some of these UIISA load instructions into two separately executable IISA instructions. These two IISA instructions are defined

5

herein as a PREFETCH instruction that, if necessary, causes specified data to be prefetched from lower level memory (e.g., L2 cache 17 or system memory 8) into L1 data cache 15 and a REGISTER instruction that transfers data specified by the UISA load instruction into a register file.

10

Referring now to **Figures 2A** and **2B**, there are depicted two alternative embodiments of the translation of UISA load instructions into separately executable PREFETCH and REGISTER instructions in accordance with the present invention. As illustrated in **Figure 2A**, in a first embodiment, ITU 18 translates UISA load instruction into two IISA LOAD instructions 40 and 42 that are identical except for the value of a register operation field 50. Thus, while LOAD instructions 40 and 42 have matching opcode, register, and address fields 44, 46 and 48, register field 50 of LOAD instruction 40 is reset to 0 to indicate a PREFETCH operation, and register field 50 of LOAD instruction 42 is set to 1 to indicate a REGISTER operation. A variation on this embodiment that could be implemented with or without instruction translation by ITU 18 would be for a single LOAD instruction to be supplied to ISU 20, and for ISU 20 to issue the LOAD instruction twice for execution (e.g., from an instruction buffer) with differing settings of register field 50.

25

30

Alternatively, as shown in **Figure 2B**, ITU 18 may translate a UISA load instruction into distinct IISA prefetch and register instructions 60 and 62, respectively. As illustrated, IISA PREFETCH instruction

5

60 contains, in addition to an opcode field 64, at least a target address field 66 identifying operands that may be utilized to compute the memory address(es) from which load data is to be retrieved. IISA REGISTER instruction 62, by contrast, has a different opcode specified in its opcode field 64 and specifies in a register field 68 the register(s) into which the load data are to be transferred.

10

0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
460
470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990

By translating UIISA instructions to IISA instructions in this manner, memory access latency associated with load instructions can be masked as in complex out-of-order machines, even in processors of reduced complexity that execute instructions either in-order or only slightly out-of-order. As an example, an exemplary cache line of instructions fetched from cache hierarchy 16 may include the following UIISA instructions:

20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
1000
1005
1010
1015
1020
1025
1030
1035
1040
1045
1050
1055
1060
1065
1070
1075
1080
1085
1090
1095
1100
1105
1110
1115
1120
1125
1130
1135
1140
1145
1150
1155
1160
1165
1170
1175
1180
1185
1190
1195
1200
1205
1210
1215
1220
1225
1230
1235
1240
1245
1250
1255
1260
1265
1270
1275
1280
1285
1290
1295
1300
1305
1310
1315
1320
1325
1330
1335
1340
1345
1350
1355
1360
1365
1370
1375
1380
1385
1390
1395
1400
1405
1410
1415
1420
1425
1430
1435
1440
1445
1450
1455
1460
1465
1470
1475
1480
1485
1490
1495
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1555
1560
1565
1570
1575
1580
1585
1590
1595
1600
1605
1610
1615
1620
1625
1630
1635
1640
1645
1650
1655
1660
1665
1670
1675
1680
1685
1690
1695
1700
1705
1710
1715
1720
1725
1730
1735
1740
1745
1750
1755
1760
1765
1770
1775
1780
1785
1790
1795
1800
1805
1810
1815
1820
1825
1830
1835
1840
1845
1850
1855
1860
1865
1870
1875
1880
1885
1890
1895
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
2055
2060
2065
2070
2075
2080
2085
2090
2095
2100
2105
2110
2115
2120
2125
2130
2135
2140
2145
2150
2155
2160
2165
2170
2175
2180
2185
2190
2195
2200
2205
2210
2215
2220
2225
2230
2235
2240
2245
2250
2255
2260
2265
2270
2275
2280
2285
2290
2295
2300
2305
2310
2315
2320
2325
2330
2335
2340
2345
2350
2355
2360
2365
2370
2375
2380
2385
2390
2395
2400
2405
2410
2415
2420
2425
2430
2435
2440
2445
2450
2455
2460
2465
2470
2475
2480
2485
2490
2495
2500
2505
2510
2515
2520
2525
2530
2535
2540
2545
2550
2555
2560
2565
2570
2575
2580
2585
2590
2595
2600
2605
2610
2615
2620
2625
2630
2635
2640
2645
2650
2655
2660
2665
2670
2675
2680
2685
2690
2695
2700
2705
2710
2715
2720
2725
2730
2735
2740
2745
2750
2755
2760
2765
2770
2775
2780
2785
2790
2795
2800
2805
2810
2815
2820
2825
2830
2835
2840
2845
2850
2855
2860
2865
2870
2875
2880
2885
2890
2895
2900
2905
2910
2915
2920
2925
2930
2935
2940
2945
2950
2955
2960
2965
2970
2975
2980
2985
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8

SUB2
PRE
REG
ADD2

5

where PRE and REG denote separately executable IISA PREFETCH and REGISTER instructions, respectively.

10 If instruction sequencing unit 20 enforces in-order execution, which is defined to mean that no instruction that changes the state of an architected register can be executed prior to an instruction preceding it in program order, processor 10 can still enjoy the chief benefits of executing load instructions out-of-order, that is, masking memory access latency, without the concomitant complexity by speculatively executing the IISA PREFETCH instruction prior to at least one instruction preceding it in program order. In this manner, cache hierarchy 16 can speculatively initiate prefetching of the load data into L1 data cache 15 to mask data access latency, while the REGISTER instruction (which alters the architected state of processor 10) is still performed in-order. Table I summarizes an exemplary execution scenario, given the IISA instruction stream discussed above and an embodiment of processor 10 in which ISU 20 is capable of dispatching and retiring two instructions per cycle.

15

20

25

TABLE I

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6	Cycle 7
ADD1	D	X	C				
PRE	D	X		pre-fetch data to L1 data cache			
SUB1		D	X	C			
MUL1		D	X	C			
MUL2			D	X	C		
ST			D	X	C		
SUB2				D	X	C	
REG				D	X	C	
ADD2					D	X	C

In the exemplary scenario depicted in Table I, at the beginning of cycle 1, ISU 20 holds all nine of the IISA instructions, for example, in a deep instruction buffer that is preferably more than one cache line of instructions in depth. In response to detecting a PRE instruction available for dispatch in the instruction buffer, ISU 20 dispatches the PRE instruction out-of-order to LSU 26, for example, concurrent with the dispatch of ADD1 to IU 24.

During cycle 2, ISU 20 also decodes and dispatches the SUB1 and MUL1 instructions to IUs 24. Meanwhile, IU 24 executes ADD1, and LSU 26 executes the PRE instruction to calculate a speculative effective address (EA) of the data to be loaded. This speculative

5

10

25

30

EA is then translated to a real address, for example, by reference to a conventional data translation lookaside buffer (TLB), and supplied to cache hierarchy 16 as a prefetch request. Thus, if the real address hits in L1 data cache 15, then no further action is taken. However, if the real address misses in L1 data cache 15, then the real address will be furnished to L2 cache 17 as a request address. In the event of a hit in L2 cache 17, L2 cache 17 will load the associated data into L1 data cache 15; however, if the real address misses in L2 cache 17, then a request containing the real address will be sourced onto data bus 12 for servicing by system memory 18 or another processor 10. Thus, execution of the PREFETCH instruction triggers prefetching of data into cache hierarchy 16 (and preferably L1 data cache 15) that is likely to be loaded into a register file in response to execution of a REGISTER instruction. This prefetching is speculative, however, in that an intervening branch instruction may redirect the execution path, resulting in the REGISTER instruction not being executed. In addition, the contents of the registers utilized to compute the EA of the load data may be updated by an instruction executed between the PRE instruction and the associated REG instruction. However, because the PRE instruction merely affects the cache contents rather than the architected state of processor 10, no corrective action need be taken in the event of mis-speculation.

30 Next, in cycle 3, ISU 20 completes the ADD1 instruction, and its result data become part of the architected state of processor 10. As further shown in Table I, the SUB1 and MUL1 instructions are executed by

IUs **24**, and the MUL2 and ST instructions are decoded and dispatched to IU **24** and LSU **26**, respectively.

Assuming that the prefetch request missed in L1 data cache **15** and hit in L2 data cache **17**, during cycle 4 a copy of the prefetch data is loaded from L2 data cache **17** into L1 data cache **15**. The MUL2 and ST instructions are also executed by an IU **24** and LSU **26**, respectively.

In addition, ISU **20** completes the SUB1 and MUL1 instructions and decodes and dispatches the SUB2 and REG instructions to an IU **24** and LSU **26**, respectively. Thus, as required by the in-order architecture of processor **10**, the REG instruction, which affects the architected state of processor **10** is dispatched, executed and completed no earlier than SUB2, the instruction preceding it in program order.

Next, in cycle 5, the MUL2 and ST instructions are completed by ISU **20**, and the SUB2 and REG instructions are executed by an IU **24** and LSU **26**, respectively. To execute the REG instruction, LSU **26** computes the EA of the load data and supplies the EA to cache hierarchy **16**, which translates the EA to a real address and determines whether the load data associated with that real address is resident in L1 data cache **15**. Because of the earlier speculative execution of the PRE instruction, in most cases the load data is resident in L1 data cache **15**, and the REG instruction can both execute and load data into one of register files **32** or **34** in the minimum data access latency permitted by cache hierarchy **16**, which in this case is a single cycle.

Thereafter, in cycle 6, the ADD2 instruction, which is dispatched in cycle 5, is executed by one of IUs 24 concurrent with the completion of the SUB2 and REG instructions by ISU 20. As illustrated, because the PRE 5 instruction speculatively prefetches the data required for the ADD2 instruction prior to execution of the REG instruction, the ADD2 instruction, which is dependent upon the load data, is permitted to execute without any latency. Finally, ISU 20 completes the ADD2 instruction 10 during cycle 7.

0
15
20
25
30
35

25

30

35

It should be evident to those skilled in the art that various modifications of the exemplary processor described herein are possible and may be desirable, depending upon other architectural considerations. For example, it may be desirable for instruction translation unit 18 to be merged into ISU 20. In addition, it may be desirable for a processor in accordance with the present invention to permit out-of-order execution of instructions other than memory access instructions (e.g., loads and stores), while requiring memory access instructions to be executed strictly in order. In general, permitting non-memory-access instructions to execute out-of-order would not introduce any additional complexity as compared to in-order execution since conventional in-order processors include logic for detecting and observing register data dependencies between instructions. Moreover, a processor in accordance with the present invention may chose to execute the PRE instruction by speculatively loading the data into buffer storage, rather than merely "priming" the cache hierarchy with a prefetch address. Buffering speculatively fetched load data in this manner is permitted even by in-order machines in that the content of the register files is not affected.

5

10

15
20
25
30

For example, **Figure 3** illustrates a load data queue **80** within **LSU 26** that may be utilized to temporarily buffer load data received from cache hierarchy **16** in response to execution of a PREFETCH instruction. As shown, each entry of load data queue **80** associates load data retrieved from cache hierarchy **16** with the target address (TA) from which the load data was retrieved and the EA of the UIISA load instruction, which is shared by and flows through processor **10** in conjunction with each of the PREFETCH and REGISTER IISA instructions. Thus, when **LSU 26** subsequently executes a REG instruction, the EA of the UIISA load instruction (and thus the IISA REG instruction) forms an index into load data queue **80** and the TA provides verification that the speculatively calculated target address was correct. Although implementing a load data queue such as that depicted in **Figure 3** may reduce access latency in some implementations, the improvement in access latency entails additional complexity in that store operations and exclusive access requests by other processors must be snooped against the load data queue to ensure correctness.

30

In another embodiment of the present invention, it may be desired to permit the PREFETCH instruction to be issued and executed as early as possible, but still constrain the PREFETCH instruction to be executed without utilizing speculative address operands. That is, when dispatching instructions, **ISU 20** would still advance the PREFETCH instruction as far as possible in execution order with respect to the REGISTER instructions, but processor **10** would enforce register data dependencies so that PREFETCH instructions would always use correct

(i.e., non-speculative) register values when computing the prefetch address.

As has been described, the present invention provides an improved processor and method of performing load operations that translate UIISA load operations into separately executable prefetch and register operations. Because performing the prefetch operation does not affect the architected state of a processor, the prefetch operation can be performed speculatively to mask data access latency, even in in-order execution machines. The register operation can thereafter be performed in-order to complete the load operation.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95

CLAIMS

What is claimed is:

1. A method of processing an instruction in a processor,
2 said method comprising:

3 fetching a sequence of instructions including
4 an instruction; and

5 translating the instruction into separately
6 executable prefetch operation and register operations,
7 wherein said prefetch operation obtains, in an out-of-
8 order fashion, data need to execute said register
operation and said register operation performs an
16 operation in order.

1 2. A processor, comprising:

2 means for fetching a sequence of instructions
3 including an instruction; and

4 means for translating the instruction into
5 separately executable prefetch operation and register
6 operations, wherein said prefetch operation obtains, in
7 an out-of-order fashion, data need to execute said
8 register operation and said register operation performs
9 an operation in order.

2 a plurality of registers;

3 instruction processing circuitry that fetches a
4 load instruction and a preceding instruction that
5 precedes said load instruction in program order and,
6 responsive to detecting said load instruction, translates
7 said load instruction into separately executable prefetch
8 and register operations; and

9 execution circuitry that performs at least said
10 prefetch operation out-of-order with respect to said
11 preceding instruction to prefetch data and subsequently
12 separately executes said register operation to place said
13 data into a register among said plurality of registers
14 specified by said load instruction.

1 4. The processor of Claim 3, wherein said execution
2 circuitry executes said register operation in-order with
3 respect to said preceding instruction.

1 5. The processor of Claim 3, wherein said execution
2 circuitry executes said register operation out-of-order
3 with respect to said preceding instruction.

1 6. The processor of Claim 3, wherein said prefetch
2 operation and said register operation have a same
3 operation code.

1 7. The processor of Claim 6, wherein said prefetch
2 operation and said register operation differ only in a
3 value of a register operation field.

1 8. The processor of Claim 3, wherein said execution
2 circuitry stores said data prefetched in response to said
3 prefetch operation in a temporary register.

1 9. The processor of Claim 3, and further comprising a
2 data hazard detector that, in response to detection of a
3 hazard for said data, signals said processor to discard
4 said data and said register operation.

1 10. A method of performing a load operation in a
2 processor having a plurality of registers, said method
3 comprising:

4 fetching a load instruction and a preceding
5 instruction that precedes said load instruction in
6 program order;

7 detecting said load instruction and translating
8 said load instruction into separately executable prefetch
9 and register operations;

10 performing at least said prefetch operation
11 out-of-order with respect to said preceding instruction
12 to prefetch data; and

13 thereafter, separately executing said register
14 operation to place said data into a register among said
15 plurality of registers specified by said load
16 instruction.

1 11. The method of Claim 10, and further comprising
2 executing said register operation in-order with respect
3 to said preceding instruction.

1 12. The method of Claim 10, and further comprising
2 executing said register operation out-of-order with
3 respect to said preceding instruction.

1 13. The method of Claim 10, wherein translating said
2 load instruction comprises translating load operation
3 into prefetch and register operation have a same
4 operation code.

1 14. The method of Claim 13, wherein said prefetch
2 operation and said register operation differ only in a
3 value of a register operation field.

1 15. The method of Claim 10, wherein performing said
2 prefetch operation comprises storing said data in a
3 temporary register.

1 16. The method of Claim 10, and further comprising:

2 detecting a data hazard for said data; and
3 in response to detection of said hazard for
4 said data, discarding said data and said register
5 operation.

ABSTRACT OF THE DISCLOSURE

PROCESSOR AND METHOD OF EXECUTING A LOAD INSTRUCTION THAT
BIFURCATE LOAD EXECUTION INTO TWO OPERATIONS

5 A processor implementing an improved method for executing load instructions includes execution circuitry, a plurality of registers, and instruction processing circuitry. The instruction processing circuitry fetches a load instruction and a preceding instruction that precedes the load instruction in program order, and in response to detecting the load instruction, translates the load instruction into separately executable prefetch and register operations. The execution circuitry performs at least the prefetch operation out-of-order with respect to the preceding instruction to prefetch data into the processor and subsequently separately executes the register operation to place the data into a register specified by the load instruction. In an embodiment in which the processor is an in-order machine, the register operation is performed in-order with respect to the preceding instruction.

10

20

20

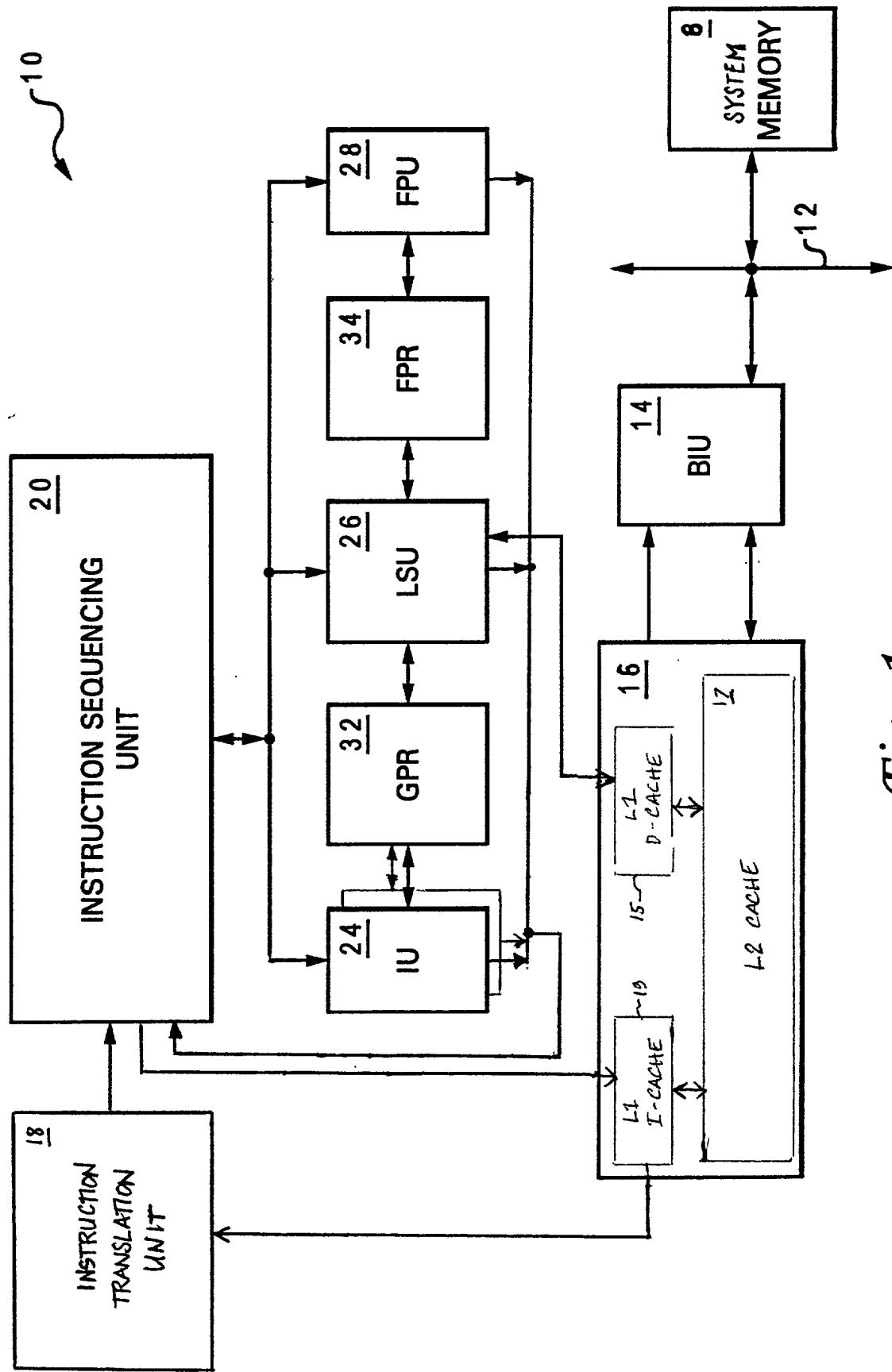


Fig. 1

AT9-99-453

2/2

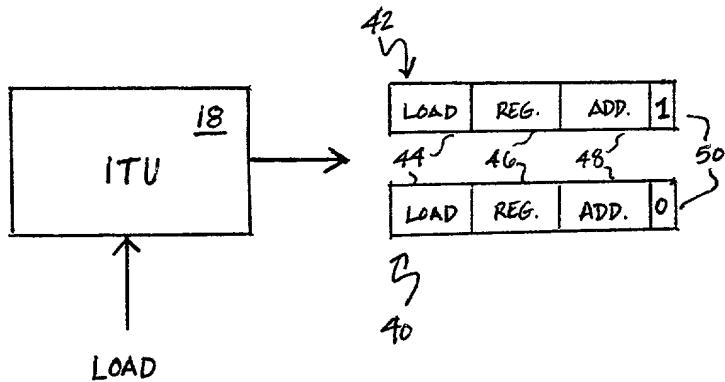


Fig. 2A

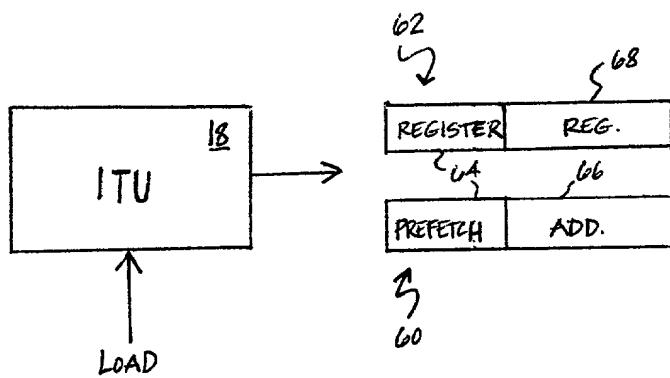


Fig. 2B

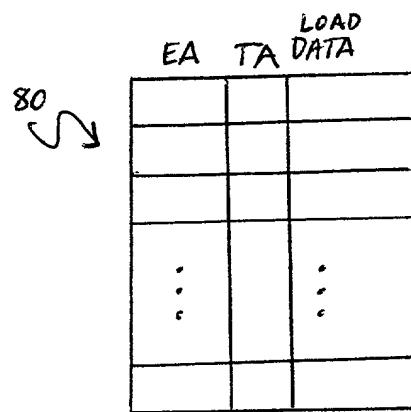


Fig. 3

DECLARATION AND POWER OF ATTORNEY FOR
PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

PROCESSOR AND METHOD OF EXECUTING A LOAD INSTRUCTION THAT BIFURCATE LOAD EXECUTION INTO TWO OPERATIONS

the specification of which (check one)

is attached hereto.

was filed on _____
as Application Serial No. _____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s): (Number)	Priority Claimed (Country)	(Day/Month/Year)	<input type="checkbox"/> Yes <input type="checkbox"/> No
---	-------------------------------	------------------	--

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal

Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial #)

(Filing Date)

(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; Thomas E. Tyson, Reg. No. 28,543; Robert M. Carwell, Reg. No. 28,499; Jeffrey S. LaBaw, Reg. No. 31,633; Douglas H. Lefeve, Reg. No. 26,193; Casimer K. Salys, Reg. No. 28,900; David A. Mims, Jr., Reg. No. 32,708; Volel Emile, Reg. No. 39,969; James H. Barksdale, Jr. Reg. No. 24,091; Anthony V. England, Reg. No. 35,129; Leslie A. Van Leeuwen, Reg. No. 42,196; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Andrew J. Dillon, Reg. No. 29,634; Max Ciccarelli, Reg. No. 39,454; Daniel E. Venglarik, Reg. No. 39,409; Jack V. Musgrove, Reg. No. 31,986; Brian F. Russell, Reg. No. 40,796; Steven Lin, Reg. No. 35,250; Matthew W. Baca, Reg. No. 42,277; Antony P. Ng, Reg. No. 43,427; John G. Graham, Reg. No. 19,563; Matthew S. Anderson, Reg. No. 39,093; Michael R. Barre, Reg. No. 44,023; Andrew Mitchell Harris, Reg. No. 42,638; Richard McCain, Reg. No. 43,785; Michael Noe, Reg. No. 44,975; and Sidney L. Weatherford, Reg. No. P-45,602.

Send correspondence to: Andrew J. Dillon, FELSMAN, BRADLEY, VADEN, GUNTER & DILLON, LLP, Suite 350 Lakewood on the Park, 7600B North Capital of Texas Highway, Austin, Texas 78731, and direct all telephone calls to Andrew J. Dillon, (512) 343-6116.

FULL NAME OF SOLE OR FIRST INVENTOR: Charles Robert MooreINVENTORS SIGNATURE: Charles Robert Moore DATE: 6/12/2000RESIDENCE: 8802 Royalwood Drive
Austin, Texas 78750CITIZENSHIP: U.S.A.POST OFFICE ADDRESS: 8802 Royalwood Drive
Austin, Texas 78750